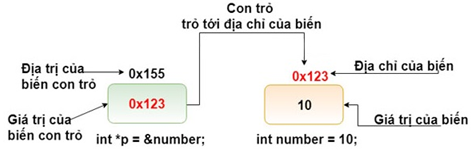
#### 1. Giới thiệu con trỏ

##### 1.2. Con trỏ là gì

Con trỏ trong C là một loại biến đặc biệt mà giá trị của nó là địa chỉ của 1 biến khác.



Một số khái niệm về con trỏ:

* Giá trị của con trỏ: địa chỉ mà con trỏ trỏ đến.
* Địa chỉ của con trỏ: địa chỉ của bản thân biến con trỏ đó.
* Giá trị của biến nơi con trỏ đang trỏ tới.
* Địa chỉ của biến nơi con trỏ đang trỏ tới = giá trị của con trỏ.

Chính vì con trỏ mang địa chỉ, nó là một biến đặc biệt có thêm những quyền năng mà biến bình thường không có. Nhờ việc nó mang địa chỉ, nó có thể trỏ lung tung trong bộ nhớ.

##### 1.3. Cách khai báo con trỏ

Con trỏ trong C cũng có thể khai báo giống như biến bình thường, tên biến là một định danh hợp lệ.

|  |
| --- |
| **Cú pháp:** |
| Kiểu dữ liệu \* Tên biến |

Trong đó:

* Kiểu dữ liệu có thể là: void, int, float, double,…
* Dấu \* trước tên biến là ký hiệu báo cho trình biên dịch biết ta đang khai báo con trỏ.

**Ví dụ:**

|  |  |
| --- | --- |
| int \*p\_i; // khai báo con trỏ để trỏ tới biến kiểu nguyên  int \*p, val; // khai báo con trỏ p kiểu int, biến val (không phải con trỏ) kiểu int  float \*p\_f; // khai báo con trỏ để trỏ tới biến kiểu thực  char \*p\_char; // khai báo con trỏ để trỏ tới biến kiểu ký tự  void \*p\_v; // con trỏ kiểu void (không kiểu) |  |

##### 1.5. Cách tạo một con trỏ

Để tạo một biến số dạng con trỏ, ta cần phải thêm kí tự \* trước tên của biến số.

|  |
| --- |
| int \*pointer; |

Ngoài ra, chúng ta cũng có thể viết: int\* pointer; vẫn hoạt động tương tự như trên. Nhưng cách viết đầu được khuyến khích hơn. Bởi vì trong trường hợp bạn cần khai báo cùng lúc nhiều con trỏ trong cùng một dòng, bạn bắt buộc phải đặt \* trước mỗi tên con trỏ: int \*pointer1, \*pointer2, \*pointer3;

Giống như điều tôi dạy bạn khi khai báo biến số, bạn cần cho nó giá trị ngay khi khởi tạo, rất quan trọng, bằng cách cho nó giá trị 0 (lấy ví dụ với biến số). Và đối với con trỏ, điều này còn quan trọng hơn nữa! Để khởi tạo con trỏ, có nghĩa là cho nó một giá trị mặc định, người ta không dùng giá trị 0 mà dùng từ khóa NULL (phải được viết hoa):

**C code:**

|  |
| --- |
| int \*pointer = NULL; |

Bạn đã khởi tạo một con trỏ giá trị *NULL*. Như vậy, bạn chắc rằng con trỏ của bạn không chứa địa chỉ nào.

##### Tạo sao lại là NULL ?

***Việc này diễn ra như thế nào?***

Đoạn mã trên sẽ đặt trước một chỗ trong bộ nhớ, giống như cách bạn tạo ra một biến số thông thường. Nhưng điều thay đổi ở đây là giá trị của con trỏ chỉ dùng để chứa địa chỉ của một biến số khác.

***Vậy thử xem với địa chỉ của tuoi thì sao?***

Và đây là cách chỉ ra địa chỉ của một biến số (*tuoi*) dựa trên giá trị của nó (bằng cách sử dụng kí tự &), nào bắt đầu thôi!

**C code:**

|  |
| --- |
| int tuoi = 10;  int \*pointerTuoi = &tuoi; |

Dòng thứ nhất : Tạo một biến số tuoi thuộc type int có giá trị là 10.

Dòng thứ hai : Tạo một con trỏ pointerTuoi có giá trị là địa chỉ của biến số tuoi.

Dòng thứ hai thực hiện cùng lúc hai việc. Nếu bạn thấy phức tạp nên không muốn gộp hai việc với nhau, tôi sẽ tách biệt chúng bằng cách chia thành hai giai đoạn.

**C code:**

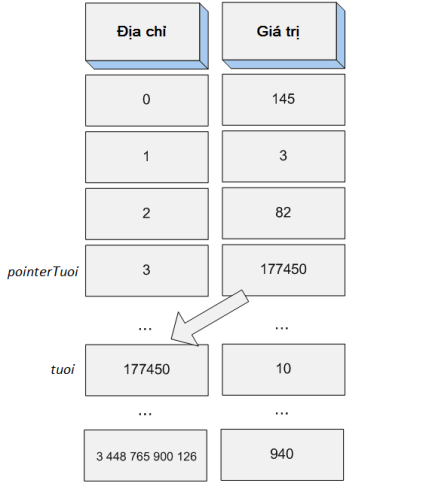
|  |
| --- |
| int tuoi = 10;  int \*pointerTuoi;    // Tạo một con trỏ poiterTuoi.  pointerTuoi = &tuoi; //Con trỏ poiterTuoi" chứa địa chỉ của biến số tuoi. |

##### Ta có thể khai báo float \*pointerTuoi; được không. Tại sao lại như vậy?

Không. Chúng ta cần phải chỉ rõ type con trỏ phù hợp với type mà con trỏ sẽ chứa địa chỉ của nó.

**VD:**

Con trỏ pointerTuoi sẽ chứa địa chỉ của biến số tuoi có type là int, vậy con trỏ phải có type cùng type với biến số tuoi là int. Nếu biến số tuoi có type là double, ta phải viết double \*pointerTuoi.



Hình 8.2: sau tóm tắt lại những gì đã diễn ra trong bộ nhớ:

Trong hình trên, biến số tuoi được đặt vào ô địa chỉ 177450 (và bạn thấy tại đó giá trị tương ứng là 10), và con trỏ pointerTuoi được đặt vào ô địa chỉ 3 và có giá trị là 177450 (177450 là địa chỉ của biến tuoi). Tất cả địa chỉ đều được chọn ngẫu nhiên, và các địa chỉ trong hình cũng do tôi tự viết ra.

|  |
| --- |
| **Đặt vấn đề:**  Ok, nhưng ... nó dùng để làm gì ? |

Ta có một con trỏ pointerTuoi chứa địa chỉ của biến số tuoi. Hãy dùng printf xem thử nó chứa gì trong đó:

**C code:**

|  |
| --- |
| int tuoi = 10;  int \*pointerTuoi = &tuoi;  printf ("%i", pointerTuoi); |

**Console**

|  |
| --- |
| 177450 |

***Vậy làm sao có được giá trị của biến số mà pointerTuoi chỉ vào?***

Chúng ta phải đặt kí tự \* trước tên của con trỏ:

**C code:**

|  |
| --- |
| int tuoi = 10;  int \*pointerTuoi = &tuoi;  printf ("%i", \*pointerTuoi); |

**Console**

|  |
| --- |
| 10 |

Bằng cách đặt kí tự \* trước tên con trỏ pointerTuoi, ta nhận được giá trị của biến số tuoi.

Nếu chúng ta sử dụng kí tự & trước tên của con trỏ, chúng ta sẽ nhận được địa chỉ của con trỏ (trong trường hợp này là 3).

|  |
| --- |
| **Câu hỏi:**  Vậy tôi đạt được điều gì ở đây? Nãy giờ tôi chỉ thấy các vấn đề càng lúc càng rắc rối thêm. Tôi thấy không cần thiết hiển thị giá trị của biến số tuoi bằng con trỏ! Trước đây chúng ta vẫn có thể hiển thị giá trị của biến số mà đâu cần đến con trỏ. |

Đây là một câu hỏi chính đáng (mà bạn bắt buộc phải đặt ra cho bản thân). Sau tất cả những điều rắc rối bạn vừa được học, hiển nhiên là bạn muốn biết tác dụng của nó, nhưng tại thời điểm này, tôi khó có thể giải thích hết, qua các bài học sau, từng chút một, các bạn sẽ thấy nó không đơn giản được tạo ra chỉ để làm mọi thứ phức tạp thêm.

Nếu bạn hiểu được nguyên lý hoạt động, thì chắc chắn, những thắc mắc sẽ được sáng tỏ trong các bài học sau.

##### 1.4. Gán giá trị cho con trỏ

Sau khi khai báo con trỏ, bạn cần khởi tạo giá trị cho nó. Nếu con trỏ được sử dụng mà không được khởi tạo, giá trị của nó sẽ là giá trị rác, điều này sẽ làm chương trình của bạn chạy không đúng, thậm chí là nguy hiểm nếu giá trị rác đó chẳng may lại chính là địa chỉ của 1 biến nào đó bạn đang dùng.

Khi khai báo con trỏ mà hiện chưa sử dụng đến thì chúng ta nên gán giá trị cho nó là NULL. Khi đó con trỏ NULL luôn có giá trị 0.

|  |
| --- |
| **VD:** Gán giá trị của con trỏ là NULL |
| #include <stdio.h>  int main () {  int  \*p;  p = NULL;  printf("\nDia chi p: %i - Gia tri cua a: %i", &p, p);  return 0;  } |
| **Kết quả:** |
|  |

Trong hầu hết các hệ điều hành, các chương trình không được phép truy cập bộ nhớ ở địa chỉ 0 vì bộ nhớ đó được hệ điều hành dành riêng. Tuy nhiên, địa chỉ bộ nhớ 0 có ý nghĩa đặc biệt; nó báo hiệu rằng con trỏ không nhằm mục đích trỏ đến một vị trí bộ nhớ có thể truy cập được. Nhưng theo quy ước, nếu một con trỏ chứa giá trị NULL (\0), nó được coi là không trỏ đến.

**VD:**

Khởi tạo con trỏ p có giá trị là địa chỉ của biến value

|  |
| --- |
| int \*p, value;  value = 5;  p = &value; |

Hoặc bạn cũng có thể khai báo và khởi tạo đồng thời:

|  |
| --- |
| int value = 5;  int \*p = &value; |

**Lưu ý:**

|  |
| --- |
| * Con trỏ khi khai báo nên được khởi tạo giá trị ngay. * Con trỏ kiểu void là loại biến con trỏ tổng quát, nó có thể nhận địa chỉ của biến bất kỳ ở bất cứ kiểu dữ liệu nào. |

Ta cần phân biệt rõ giữa khai báo con trỏ, giá trị của con trỏ và giá trị của biến mà con trỏ đang trỏ đến.

**Ví dụ:**

|  |
| --- |
| **// Khai báo, Khởi tạo con trỏ p có giá trị là địa chỉ biến a**  int \*p, a = 5;  p = &a;  **// Vừa khai báo và khởi tạo con trỏ p có giá trị là địa chỉ biến a**  int a = 5;  int \*p = &a;  **// (Giá trị của biến mà con trỏ p đang trỏ tới) gán = 5**  \*p = 5 |

**VD:Xác**

|  |
| --- |
| #include <stdio.h>  int main() {  int a = 5, b = 8;  int \*p; **// Khai báo con trỏ p**  p = &a; **// Giá trị của con trỏ p**  printf("\nDia chi a: %i - Gia tri cua a: %i", &a, a);  printf("\nDia chi p: %i - Gia tri cua p: %i", &p, p);  printf("\nDia chi p: %i - Gia tri cua p: %i", &p, \*p);  \*p = b; **//\*p là lấy giá trị của biến mà con trỏ đang trỏ đến.**  printf("\nDia chi p: %i - Gia tri cua p: %i", &p, p);  printf("\nDia chi p: %i - Gia tri cua p: %i", &p, \*p);  printf("\nDia chi a: %i - Gia tri cua a: %i", &a, a);  } |

**Kết quả:**

|  |
| --- |
| Dia chi a: 6487576 - Gia tri cua a: 5  Dia chi p: 6487568 - Gia tri cua p: 6487576  Dia chi p: 6487568 - Gia tri cua p: 5  Dia chi p: 6487568 - Gia tri cua p: 6487576  Dia chi p: 6487568 - Gia tri cua p: 8  Dia chi a: 6487576 - Gia tri cua a: 8 |

##### 1.4. Cách sử dụng con trỏ

Đến bây giờ, bạn chỉ có thể tạo biến số để chứa các số hạng. Và sau đây chúng ta sẽ học cách tạo ra những biến số chứa địa chỉ của chúng, những biến số này gọi là con trỏ.

|  |
| --- |
| **Đặt vấn đề:**  Nhưng, địa chỉ cũng là một số đúng không? Như vậy số này cần phải chứa trong một biến số khác và cứ như thế nó sẽ lặp lại mãi sao? |

Chính xác. Nhưng các con số này sẽ có một kí hiệu đặc biệt để nhận biết: địa chỉ của một biến số khác trong bộ nhớ.

##### 1.5. Bản chất của con trỏ trong C

Bạn sẽ hiểu rõ hơn các quyền năng của con trỏ trong phần này, cũng xem ví dụ dưới đây nào:

|  |
| --- |
| #include <stdio.h>  int main() {  **// Khai báo + khởi tạo biến value = 10**  int value = 10;  **// Lấy giá trị của biến value**  printf("\nGia tri cua \"value\" = %i", value);  **// Lấy địa chỉ của biến value**  printf("\nDia tri cua \"value\" = %i", &value);  printf("\n-------------------\n");    **/\* Khai báo + khởi tạo biến con trỏ p có giá trị là địa chỉ của biến value \*/**  int \*p = &value;  **// Lấy giá trị của con trỏ p**  printf("\nGia tri cua con tro \"p\" = %i", p);  **// Lấy địa chỉ của con trỏ p**  printf("\nDia tri cua con tro \"p\"  = %i", &p);  **// Lấy giá trị của biến mà con trỏ p đang trỏ tới dùng toán tử \***  printf("\nGia tri cua bien ma con tro \"p\" dang tro toi = %i", \*p);  printf("\n-------------------\n");    **/\*Thay đổi giá trị của biến value thông qua con trỏ p. Giống như hàm scanf() có thể thay đổi giá trị của biến khi nhận vào địa chỉ, con trỏ khi có địa chỉ của 1 biến hoàn toàn có thể thay đổi giá trị của**  **biến đó theo cách dưới đây:\*/**  **// Lấy giá trị của biến value**  printf("\nGia tri cua \"value\" = %i", value);  **// Thay đổi giá trị của biến value thông qua con trỏ p**  \*p = 100;  **// Lấy giá trị của biến value**  printf("\nGia tri cua \"value\" = %i", value);  **// Lấy giá trị của biến mà con trỏ p đang trỏ tới dùng toán tử \***  printf("\nGia tri cua bien ma con tro \"p\" dang tro toi = %i", \*p);  printf("\n-------------------\n");    **/\*Việc lấy giá trị của biến thông qua con trỏ chỉ là 1 cách khác để lấy được giá trị của biến đó.\*/**  value = 1000;  **// Lấy giá trị của biến value**  printf("\nGia tri cua \"value\" = %i", value);  **// Lấy giá trị của biến mà con trỏ p đang trỏ tới dùng toán tử \***  printf("\nGia tri cua bien ma con tro \"p\" dang tro toi = %i", \*p);  } |

**Kết quả:**

|  |
| --- |
| Gia tri cua "value" = 10  Dia tri cua "value" = 6487580  -------------------  Gia tri cua con tro "p" = 6487580  Dia tri cua con tro "p" = 6487568  Gia tri cua bien ma con tro "p" dang tro toi = 10  -------------------  Gia tri cua "value" = 10  Gia tri cua "value" = 100  Gia tri cua bien ma con tro "p" dang tro toi = 100  -------------------  Gia tri cua "value" = 1000  Gia tri cua bien ma con tro "p" dang tro toi = 1000 |

**Nhận xét:**

|  |
| --- |
| * Địa chỉ của biến value chính là giá trị của con trỏ p, đều là 6487580. Lưu ý mỗi lần chạy thì giá trị địa chỉ này có thể khác nhau. * Con trỏ có thể lấy giá trị của biến mà nó đang trỏ tới bằng toán tử \*:   printf("\nGia tri cua bien ma con tro \"p\" dang tro toi = %i", \*p);   * Con trỏ có thể thay đổi giá trị của biến mà nó đang trỏ tới. Do nó mang địa chỉ của biến, khi đó nó hoàn toàn có quyền thay đổi giá trị của biến đó. Như ở ví dụ trên ta thay đổi giá trị từ 10 lên 100. |

#### 2. Các lỗi thường gặp khi làm việc với con trỏ

Giả sử bạn muốn khởi tạo giá trị của con trỏ p trỏ tới địa chỉ của biến value, khi đó:

|  |
| --- |
| int value, \*p;    **/\* Sai.**   * **p cần 1 địa chỉ.** * **value không phải là cái địa chỉ. \*/**   p = value;    **/\* Sai.**   * **\*p là giá trị của biến mà con trỏ đang trỏ tới.** * **// &value là địa chỉ. \*/**   \*p = &value;    **/\* Đúng.**   * **p cần 1 địa chỉ.** * **// &value là địa chỉ của biến value.**   p = &value;    **/\* Đúng.**   * **\*p là giá trị của biến mà con trỏ đang trỏ tới.** * **value cũng là giá trị (không phải địa chỉ). \*/**   \*p = value; |

Chú ý về dấu \* ở phần khai báo và khi lấy giá trị của biến mà con trỏ đang trỏ tới:

|  |
| --- |
| #include <stdio.h>  int main() {      int c = 5;  **// Dấu \* ở đây để chúng ta biết chúng ta đang khai báo con trỏ.**  **// Không phải lấy giá trị của nó nhé**      int \*p = &c;      // Khai báo trên tương đương      // int \*p;      // p = &c;      // Nếu bạn muốn phân biệt 2 thằng này, khi khai báo có thể viết như sau:      // int\* p = &c;        // Lấy giá chỉ của biến mà con trỏ đang trỏ tới, chính là giá trị của c      printf("%i", \*p); // 5  } |

#### 3. Con trỏ Null

Việc gán giá trị NULL cho một biến con trỏ luôn là một thực tiễn tốt trong trường hợp bạn không có địa chỉ chính xác để gán. Điều này được thực hiện tại thời điểm khai báo biến. Một con trỏ được gán NULL được gọi là con trỏ null .

Con trỏ NULL là một hằng số có giá trị bằng 0 được xác định trong một số thư viện chuẩn. Hãy xem xét chương trình sau:

|  |
| --- |
| #include <stdio.h>  int main () {      int  \*ptr = NULL;      printf("Gia tri cua con tro NULL: %x\n", ptr  );      return 0;  } |

Khi đoạn mã trên được biên dịch và thực thi, nó tạo ra kết quả sau:

|  |
| --- |
| Gia tri cua con tro NULL: 0 |

Trong hầu hết các hệ điều hành, các chương trình không được phép truy cập bộ nhớ ở địa chỉ 0 vì bộ nhớ đó được hệ điều hành dành riêng. Tuy nhiên, địa chỉ bộ nhớ 0 có ý nghĩa đặc biệt; nó báo hiệu rằng con trỏ không nhằm mục đích trỏ đến một vị trí bộ nhớ có thể truy cập được. Nhưng theo quy ước, nếu một con trỏ chứa giá trị null (không), nó được coi là không trỏ tới.

Để kiểm tra con trỏ null, bạn có thể sử dụng câu lệnh 'if' như sau:

if(ptr) /\* Nó không phải là con trỏ NULL \*/

if(!ptr) /\* Nó là con trỏ NULL \*/

### BÀI 2: CÁC LOẠI CON TRỎ TRONG C

Con trỏ có nhiều khái niệm nhưng dễ và chúng rất quan trọng đối với lập trình C. Các khái niệm con trỏ quan trọng sau đây nên rõ ràng đối với bất kỳ lập trình viên C nào:

|  |  |
| --- | --- |
| Sr.No. | Khái niệm & Mô tả |
| 1 | ***Con trỏ số học***  Có bốn toán tử số học có thể được sử dụng trong con trỏ: ++, -, +, - |
| 2 | ***Mảng con trỏ***  Bạn có thể xác định mảng để chứa một số con trỏ. |
| 3 | ***Con trỏ tới con trỏ***  C cho phép bạn có con trỏ trên một con trỏ, v.v. |
| 4 | ***Chuyển con trỏ đến các hàm trong C***  Truyền một đối số theo tham chiếu hoặc theo địa chỉ cho phép đối số đã truyền được thay đổi trong hàm gọi bởi hàm được gọi. |
| 5 | ***Con trỏ trả về từ các hàm trong C***  C cho phép một hàm trả về một con trỏ tới biến cục bộ, biến tĩnh và cả bộ nhớ được cấp phát động. |

#### 1. Con trỏ số học

Một con trỏ trong C là một địa chỉ, là một giá trị số. Do đó, bạn có thể thực hiện các phép toán số học trên một con trỏ giống như bạn có thể thực hiện trên một giá trị số. Có bốn toán tử số học có thể được sử dụng trên con trỏ: ++, --, +, –

##### 1.1. Tăng một con trỏ

**VD:**Tăng biến con trỏ để truy cập từng phần tử kế tiếp của mảng.

|  |
| --- |
| #include <stdio.h>  const int MAX\_SIZE = 3;  int main () {  int  var[] = {10, 100, 200};  int  i, \*p;  p = var; **//hoặc p = var[0];**  for ( i = 0; i < MAX\_SIZE; i++) {  printf("Dia chi cua var[%i] = %x  - Gia tri cua var[%i] = %i\n", i, p, i, \*p );  p++;  }  return 0;  } |

**Kết quả:**

|  |
| --- |
| Dia chi cua var[0] = 62fe00 - Gia tri cua var[0] = 10  Dia chi cua var[1] = 62fe04 - Gia tri cua var[1] = 100  Dia chi cua var[2] = 62fe08 - Gia tri cua var[2] = 200 |

##### 1.2. Giảm một con trỏ

**VD:**Giảm biến con trỏ để truy cập từng phần tử kế tiếp của mảng.

|  |
| --- |
| #include <stdio.h>  const int MAX\_SIZE = 3;  int main () {  int  var[] = {10, 100, 200};  int  i, \*p;  p = &var[MAX\_SIZE - 1]; **// p = &var[2]**  for (i = MAX\_SIZE-1; i >= 0; i--) {  printf("Dia chi cua var[%i] = %x - Gia tri cua var[%i] = %i\n", i, p, i, \*p );  p--;  }  return 0;  } |

**Kết quả:**

|  |
| --- |
| Dia chi cua var[2] = 62fe08 - Gia tri cua var[2] = 200  Dia chi cua var[1] = 62fe04 - Gia tri cua var[1] = 100  Dia chi cua var[0] = 62fe00 - Gia tri cua var[0] = 10 |

##### 1.3. So sánh con trỏ

Các con trỏ có thể được so sánh bằng cách sử dụng các toán tử quan hệ, chẳng hạn như ==, &&, …

Nếu p1 và p2 trỏ đến các biến có liên quan đến nhau, chẳng hạn như các phần tử của cùng một mảng, thì p1 và p2 có thể được so sánh một cách có ý nghĩa.

Chương trình sau đây sửa đổi ví dụ trước - một bằng cách tăng con trỏ biến miễn là địa chỉ mà nó trỏ đến nhỏ hơn hoặc bằng địa chỉ của phần tử cuối cùng của mảng, là & var [MAX - 1]

**VD:**Tăng con trỏ biến để truy cập từng phần tử kế tiếp của

|  |
| --- |
| #include <stdio.h>  const int MAX\_SIZE = 3;  int main () {  int  var[] = {10, 100, 200};  int  i, \*p;  p = var;  i = 0;  while ( p <= &var[MAX - 1] ) {  printf("Dia chi cua var[%i] = %x - Gia tri cua var[%i] = %i\n", i, p, i, \*p );  p++;  i++;  }  return 0;  } |

**Kết quả:**

|  |
| --- |
| Dia chi cua var[0] = 62fe00 - Gia tri cua var[0] = 10  Dia chi cua var[1] = 62fe04 - Gia tri cua var[1] = 100  Dia chi cua var[2] = 62fe08 - Gia tri cua var[2] = 200 |